Machine Learning

Classification - I

Claudio Sartori

DISI
Department of Computer Science and Engineering – University of Bologna, Italy
claudio.sartori@unibo.it

Unsupervised Classification

- the unsupervised mining techniques which can be in some way related to classification are usually known in literature with names different from classification
- for this reason in this course with the term *classification* we will always mean *supervised classification*

Supervised Classification 1/2

In the following, simply *classification*

Consider the "soybean" example shown in the introduction (link to the dataset)

- The data set $\mathcal{X}$ contains N *individuals* described by D attribute values each
- We have also a $\mathcal{Y}$ vector which, for each individual x contains the class value $y(x)$
- The class allows a finite set of different values (e.g. the diseases), say C
- The class values are provided by experts: the supervisors

Supervised Classification 2/2

- We want to learn how to guess the value of the $y(x)$ for individuals which have not been examined by the experts
- We want to learn a *classification model*

Classification model

- An algorithm which, given an individual for which the class is not known, computes the class
- The algorithm is *parametrized* in order to optimize the results for the specific problem at hand
- Developing a classification model requires
  - choose the *learning algorithm*
  - let the algorithm learn its parametrization
  - assess the quality of the classification model
- The classification model is used by a run–time *classification algorithm* with the developed parametrization

Classification model or, shortly, *classifier*

A bit of formality

- a decision function which, given a data element x whose class label $y(x)$ is unknown, makes a prediction as

$$\mathcal{M}(x, \theta) = y(x)_{pred}$$

  where θ is a set of values of the parameters of the decision function
  - the prediction can be true or false
- the learning process for a given classifier $\mathcal{M}(.,.)$, given the dataset $\mathcal{X}$ and the set of supervised class labels $\mathcal{Y}$ determines θ in order to reduce the prediction error as much as possible

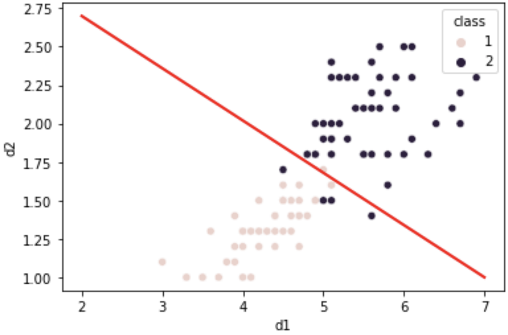
Example of decision function

- supervised dataset with two dimensions, two classes
- use as decision function a straight line

$$\theta_1 * d_1 + \theta_2 * d_2 + \theta_3 \geqslant 0 \Rightarrow c_1$$

$$\theta_1 * d_1 + \theta_2 * d_2 + \theta_3 < 0 \Rightarrow c_2$$

- this is one of the potentially infinite decision functions, θ_i are the parameters



All models are wrong, but some are useful

George Box

- The model (decision function) of the previous page makes some errors
  - even the best choice of parameters cannot avoid errors
- Different models can have different power to shatter the dataset into subsets with homogeneous classes
  - e.g. what about a quadratic function?
    $\theta_1 * d_1^2 + \theta_2 * d_2^2 + \theta_3 * d_1 * d_2 + \theta_4 * d_1 + \theta_5 d_2 + \theta_6$

Vapnik-Chervonenkis Dimension

The shattering power of a classification model[1]

- Given a dataset with N elements there are 2^N possible different learning problems
- If a model $\mathcal{M}(.,.)$ is able to shatter all the possible learning problems with N elements, we say that it has *Vapnik-Chervonenkis Dimension* equal to N
- The straight line has VC dimension 3
  - don't worry, frequently, in real cases, data are arranged in such a way that also a straight line is not so bad

1 For simplicity, we mention here only the binary case

A workflow for classification - I

1. Learning the model for the given set of classes
   1.1 a *training set* is available, containing a number of individuals
   1.2 for each individual the value of the class label is available (also named *ground truth*)
   1.3 the training set should be *representative* as much as possible
       1.3.1 the training set should be obtained by a random process
   1.4 the model is fit learning from data the best parameter setting

A workflow for classification - II

1. Estimate the *accuracy* of the model
   1.1 a *test set* is available, for which the class labels are known
   1.2 the model is run by a *classification algorithm* to assign the labels to the individuals
      1.2.1 the classification algorithm implements the model with the parameters
   1.3 the labels assigned by the model are compared with the true ones, to estimate the accuracy
2. The model is used to label new individuals
   2.1 possibly, after the labeling, the true labels may become available and the true accuracy can be compared with the estimated one

A workflow for Learning and Estimation

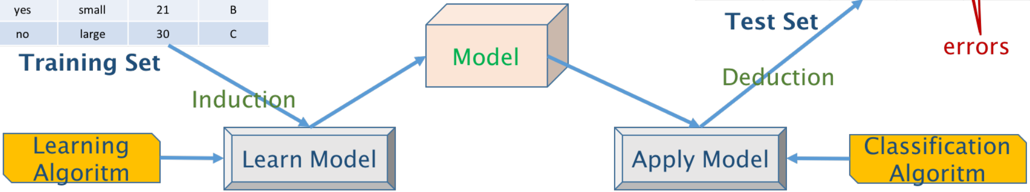
attribute 1	attribute 2	attribute 3	class
yes	large	24	A
no	medium	31	B
no	large	30	A
yes	large	22	A
yes	small	28	C
no	medium	25	C
no	small	20	A
yes	small	21	B
no	large	30	C

Training Set

attribute 1	attribute 2	attribute 3	true class	assigned class
no	small	22	A	A
no	large	30	C	C
yes	large	30	A	B
no	large	26	B	C
yes	small	23	C	C
yes	medium	25	B	B
no	medium	30	A	A

Test Set

errors



Question

Optional

- *is there a hidden assumption in the description of the soybean example of page 4?*
- *is there a workaround to this hidden assumption?*

Two flavors for classification

Crisp

- the classifier assigns to each individual *one label*

Probabilistic

- the classifier assigns a *probability for each of the possible labels*

Decision Trees

C4.5 and beyond [Buntine(1992)]

- Among the most used tools
- History
  - 1966 – ID3 [Hunt et al.(1966)Hunt, Marin, and Stone]
  - 1979 – CLS [Quinlan(1979)]
  - 1993 – C4.5 [Quinlan(1979)]
- Generate classifiers structured as *decision trees*

Using a Decision Tree 1/2[2]

- A run–time classifier structured as a decision tree is a tree–shaped set of tests
- the decision tree has *inner nodes* and *leaf nodes*

2 Syntetic description in [Wu et al.(2008)Wu, Kumar, Quinlan, Ghosh, Yang, Motoda, McLachlan, Ng, Liu, Yu, Zhou, Steinbach, Hand, and Steinberg]

Using a Decision Tree 2/2

Inner nodes:

if test on attribute *d* of element *x* **then**

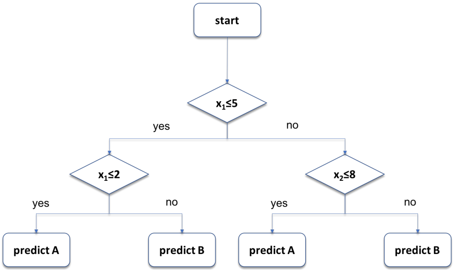
execute node'

else

execute node''

Leaf nodes:

predict class of element *x* as *c''*



Learning a decision tree – Model generation

Given a set $\mathcal{X}$ of elements for which the class is known, grow a decision tree as follows

- if all the elements belong to class c or $\mathcal{X}$ *is small* generate a leaf node with label c
- otherwise
  - choose a test based on a single attribute with two or more outcomes
  - make this test the root of a tree with one branch for each of the outcomes of the test
  - partition $\mathcal{X}$ into subsets corresponding to the outcomes and apply recursively the procedures to the subsets

Learning a decision tree

Problems to solve:

1. which attribute should we test?
2. which kind of test?
    2.1 binary, multi–way, . . . , depends also on the domain of the attribute
3. what does it mean $\mathcal{X}$ *is small*, in order to choose if a leaf node is to be generated also if the class in $\mathcal{X}$ is not unique?

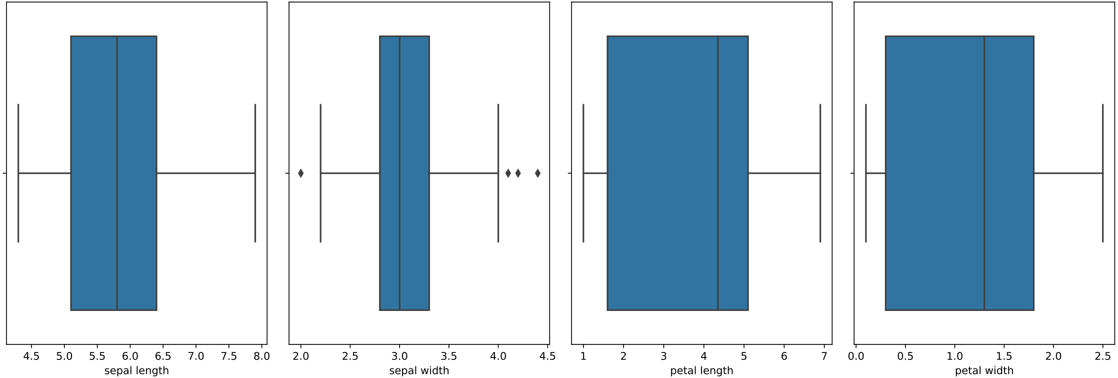
A supervised dataset: Iris

sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
6.2	2.2	4.5	1.5	1
5.2	3.5	1.5	0.2	0
5.6	3.0	4.5	1.5	1
6.0	2.9	4.5	1.5	1
7.7	3.0	6.1	2.3	2
5.1	3.8	1.5	0.3	0
5.9	3.2	4.8	1.8	1
5.7	4.4	1.5	0.4	0
6.7	3.1	5.6	2.4	2
6.5	3.2	5.1	2.0	2
...	...	...	...	...

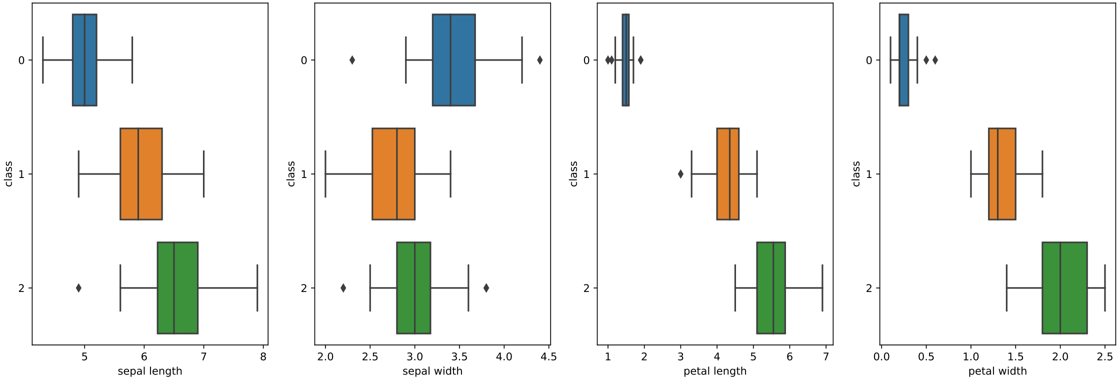
Dataset description

- 150 examples of iris flowers
- 4 attributes describing sizes of petals and sepals, class is the target
  - class has three values
- we could be interested in predicting the class for a new individual, given the measures

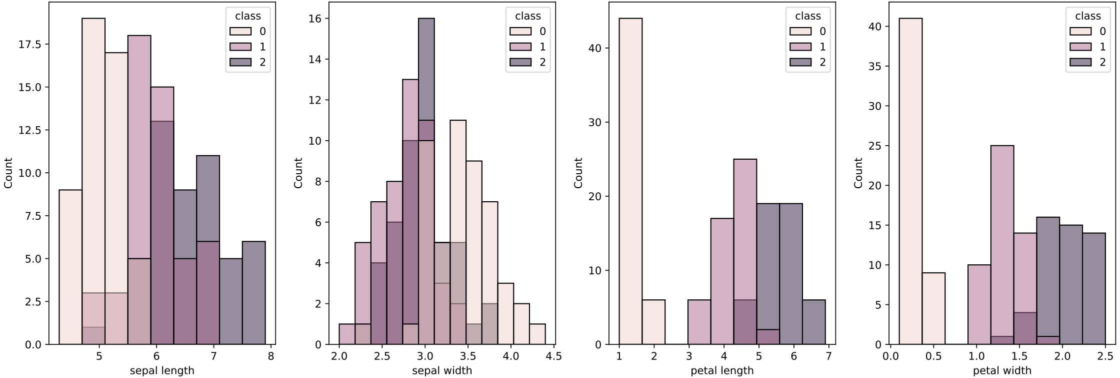
Exploration of the dataset - Boxplot - General



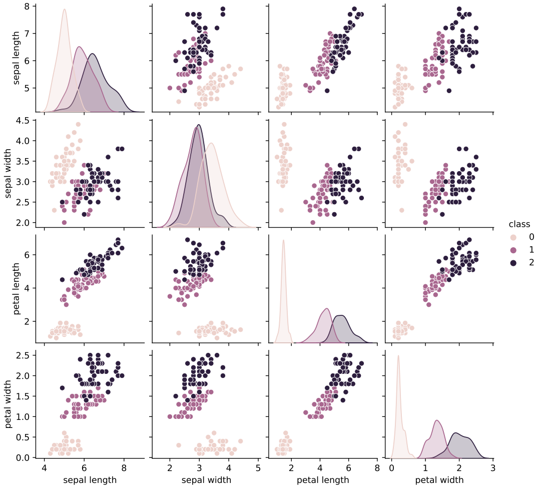
Exploration of the dataset - Boxplot - Inside classes



Exploration of the dataset - Histograms



Exploration of the dataset - Pairplots



Supervised learning goals

- design an algorithm to find interesting patterns, in order to forecast the values of an attribute given the values of other attributes
  - in our case, find patterns to guess the class given the other values
- distinguish real patterns from illusions
- choose useful patterns
- in real life, we could have millions of rows and thousands of columns
  - looking at plots could be very hard

Evaluate how much a pattern is interesting

- several methods, one of them is based on *information theory*
  - information theory is primarily used in telecommunications
  - it is based on the concept of *entropy*
    - information content, surprise, ...
    - Claude Shannon, “A Mathematical Theory of Communication”, 1948

The bit transmission example

- given a variable with 4 possible values and a given probability distribution
  $P(A) = 0.25, P(B) = 0.25, P(C) = 0.25, P(D) = 0.25$
- an observation of the data stream could return
  BAACBADCDADDDA ...
- the observation could be transmitted on a serial digital line with a two–bit coding
  $A = 00, B = 01, C = 10, D = 11$
- the transmission will be
  0100001001001110110011111100 ...

Less bits

- What if the probability distributions are uneven?
  $P(A) = 0.5, P(B) = 0.25, P(C) = 0.125, P(D) = 0.125$
- of course, the coding shown above is possible, requiring two bits per symbol
- is there a coding requiring only 1.75 bit per symbol, on the average?

Less bits

- What if the probability distributions are uneven?
  $P(A) = 0.5, P(B) = 0.25, P(C) = 0.125, P(D) = 0.125$
- of course, the coding shown above is possible, requiring two bits per symbol
- is there a coding requiring only 1.75 bit per symbol, on the average?

$$A = 0, B = 10, C = 110, D = 111$$

Even less bits

- What if there are only three symbols with equal probability? $P(A) = 1/3, P(B) = 1/3, P(C) = 1/3$
- of course, the two–bit coding shown above is still possible
- is there a coding requiring less than 1.6 bit per symbol, on the average?

Even less bits

- What if there are only three symbols with equal probability?
  $P(A) = 1/3, P(B) = 1/3, P(C) = 1/3$
- of course, the two–bit coding shown above is still possible
- is there a coding requiring less than 1.6 bit per symbol, on the average?

$$A = 0, B = 10, C = 11 \text{ or any permutation of the assignment}$$

General case

- Given a source X with V possible values, with probability distribution

$$P(v_1) = p_1, P(v_2) = p_2, \ldots, P(v_V) = p_V$$

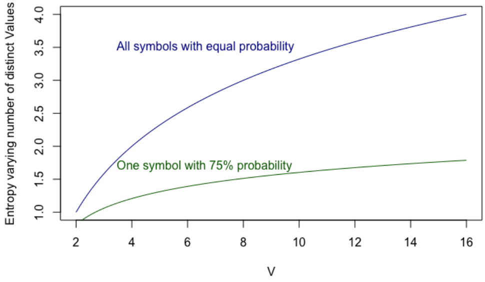
- the best coding allows the transmission with an average number of bits given by

$$H(X) = -\sum_j p_j \log_2(p_j)$$

$H(X)$ is the *entropy* of the information source X

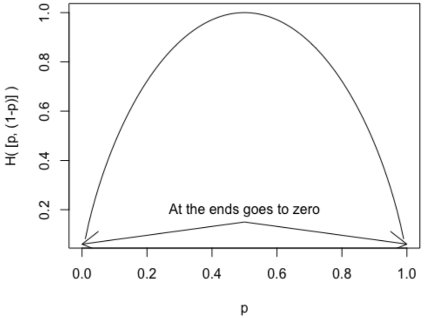
Meaning of entropy of an information source

- high entropy means that the probabilities are mostly similar
  - the histogram would be *flat*
- low entropy means that some symbols have much higher probability
  - the histogram would have *peaks*
- higher number of allowed symbols (i.e. of distinct values in an attribute) gives higher entropy



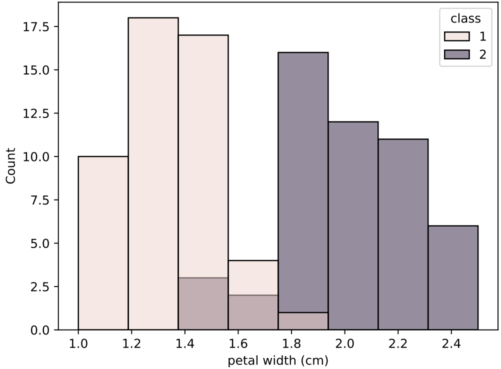
Entropy of a binary source

In a binary source with symbol probabilities p and (1-p) when p is 0 or 1 the entropy goes to 0



Entropy for the target column **class** in the reduced Iris dataset

A subset: only the fourth data column and the target, only the rows with classes 1 or 2



petal width (cm)	class
2	2
1.7	1
1.3	1
2.2	2
1.5	1
1.5	2
2.3	2
2	2
2.5	2
1.7	2
...	...

$$N = 100 \qquad p_{class=1} = 0.5, p_{class=2} = 0.5$$

$$H_{class} = -(p_{class=1} * log_2(p_{class=1}) + p_{class=2} * log_2(p_{class=2})) = 1$$

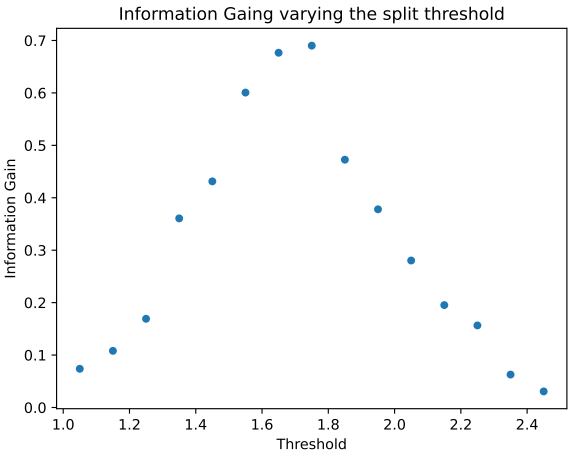
Entropy after a threshold–based split

- Splitting the dataset in two parts according to a threshold on a numeric attribute the entropy changes, and becomes the weighted sum of the entropies of the two parts
  - the weights are the relative sizes of the two parts
- Let $d \in \mathcal{D}$ be a real–valued attribute, let t be a value of the domain of d, let c be the class attribute
- We define the entropy of c w.r.t. d with threshold t as
  $H(c|d:t) = H(c|d<t) * P(d<t) + H(c|d \geqslant t) * P(d \geqslant t)$

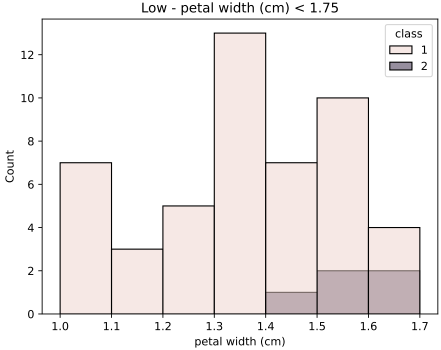
Information Gain for binary split

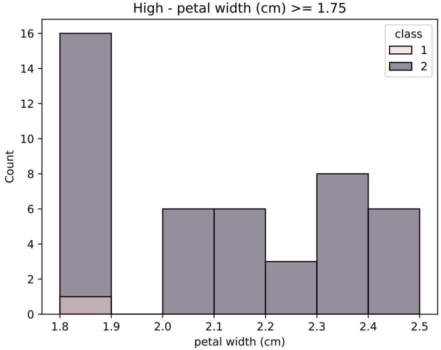
- It is the reduction of the entropy of a target class obtained with a split of the dataset based on a threshold for a given attribute
- We define $IG(c|d:t) = H(c) - H(c|d:t)$
  - it is the information gain provided when we know if, for an individual, d exceeds the threshold t in order to forecast the class value
- We define $IG(c|d) = \max_t IG(c|d:t)$

Change the threshold to find the best split



Let's split the reduced Iris with threshold 1.75





	low	high
1	49	1
2	5	45
	54	46

$$H(class|petalwidth : 1.75) = 0.31 =$$
$$- \quad (49/54 * log_2(49/54) + 5/54 * log_2(5/54)) * 0.54+$$
$$(1/46 * log_2(1/46) + 45/46 * log_2(45/46)) * 0.46$$

How can we use the information gain?

Predict the probability of long life given some historical data on person characteristics and life style

- $IG(LongLife|HairColor) = 0.01$
- $IG(LongLife|Smoker) = 0.2$
- $IG(LongLife|Gender) = 0.25$
- $IG(LongLife|LastDigitSSN) = 0.00001$

Correlations between attributes is an important issue: it is not considered here

Back to DT generation

Choosing the attribute to test

A *decision tree* is a tree–structured plan generating a sequence of tests on the known attributes (*predicting attributes*) to predict the values of an unknown attribute.

Consider question 1 of page 22: *which attribute should we test?*

- test the attribute which guarantees the maximum IG for the class attribute in the current data set $\mathcal{X}$
- partition $\mathcal{X}$ according to the test outcomes
- recursion on the partitioned data

Train/Test split[4]

- The supervised data set will be split in (at least) two parts:
  - Training set used to learn the model
  - Test set used to evaluate the learned model on fresh data
- The split is done randomly
- Assumption: the parts have similar characteristics
- The proportion of the split is decided by the experimenter
  - Common solutions: 80-20, 67-33, 50-50
- The following slides consider a 50-50 split of the Iris dataset[3]
  - For this specific split, entropies for the class column in training and test turns out to be both 1.58

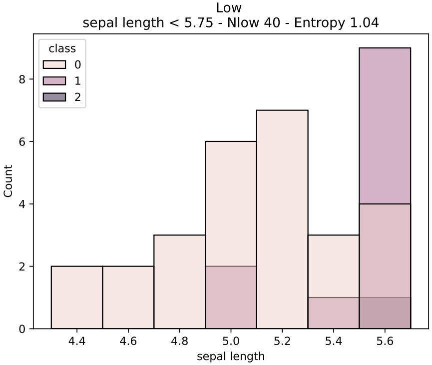
3 In the example, the split has been done using sklearn.model_selection_train_test_split and random_state = 10

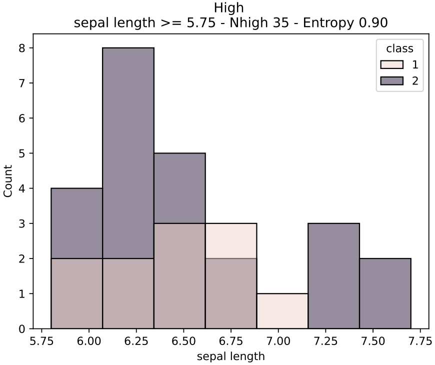
4 You can read this link for a short discussion

Iris Dataset - Predicting attribute: Sepal Length

Best threshold: 5.75

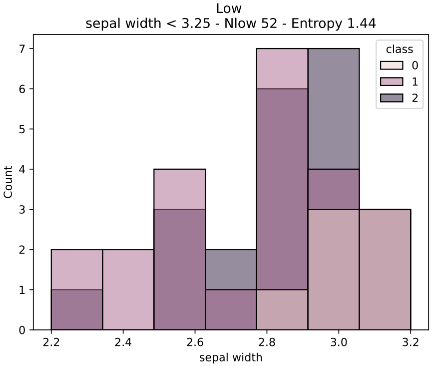
Information Gain: $1.58 - (40 * 1.04 + 35 * 0.90)/75 = 0.61$

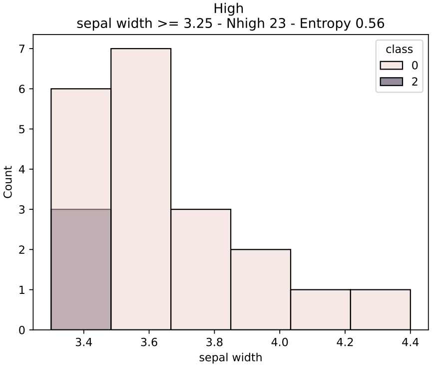




Iris Dataset - Predicting attribute: Sepal Width

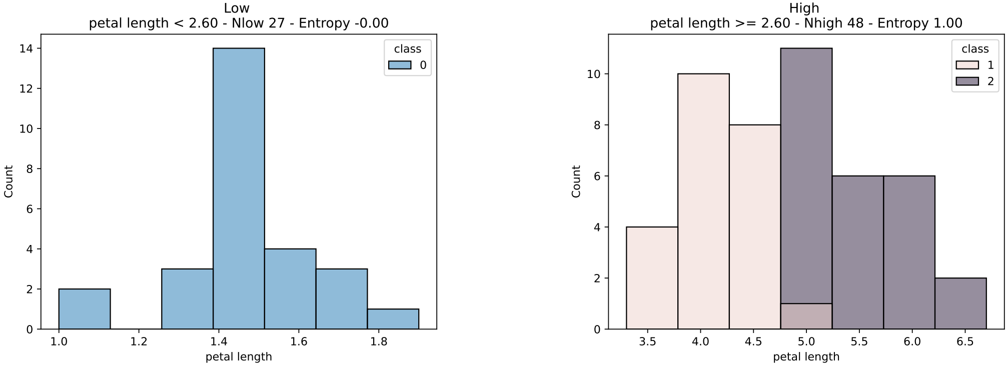
Best threshold: 3.25 – Information Gain: 1.58 -(52*1.44+23*0.56)/75 = 0.41





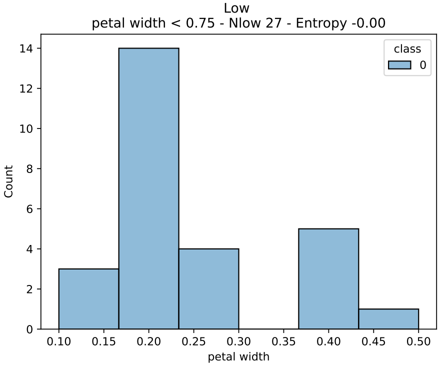
Iris Dataset - Predicting attribute: Petal Length

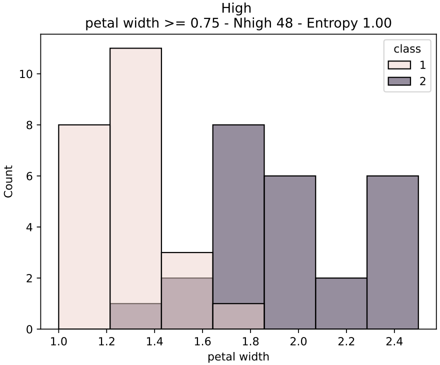
Best threshold: 2.6 – Information Gain: 0.94



Iris Dataset - Predicting attribute: Petal Width

Best threshold: 0.75 – Information Gain: 0.94

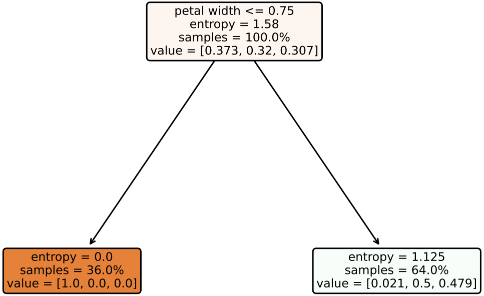




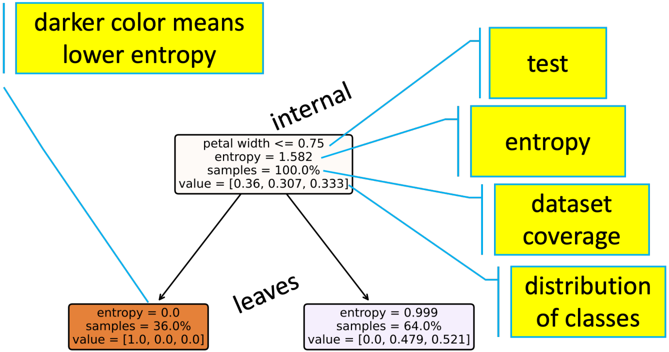
One–stump decision

Now on the entire training set with the three classes

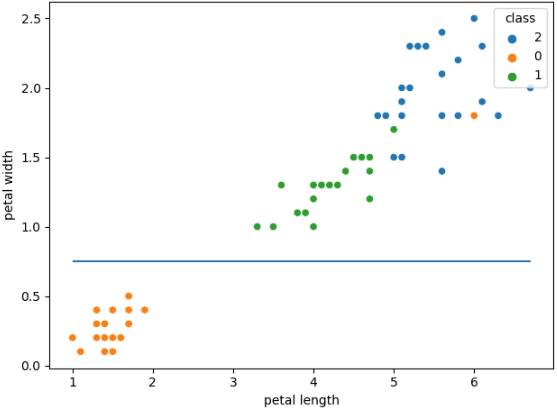
- choose the attribute giving the highest IG
- partition the dataset according to the chosen attribute
- choose as class label of each partition the majority



What's in a node

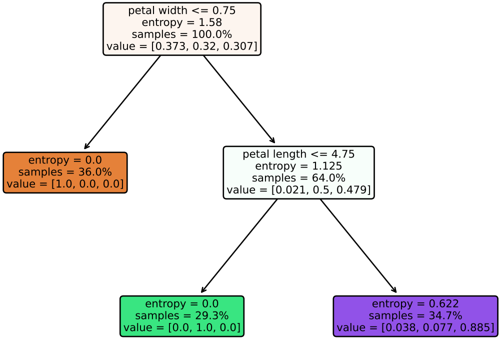


First split

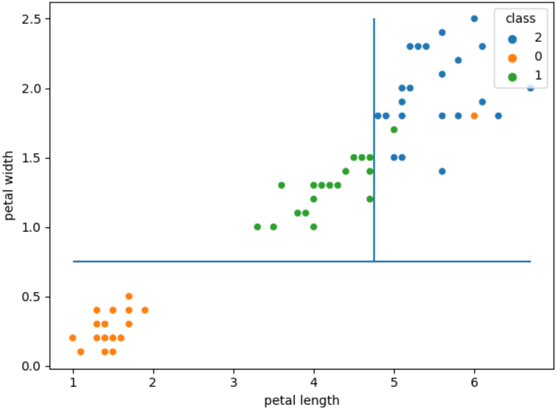


Recursion step

Build a new tree starting from each subset where the minority is non–empty

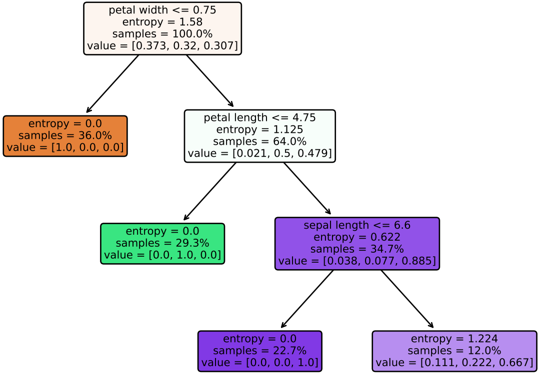


Second split



Recursion step

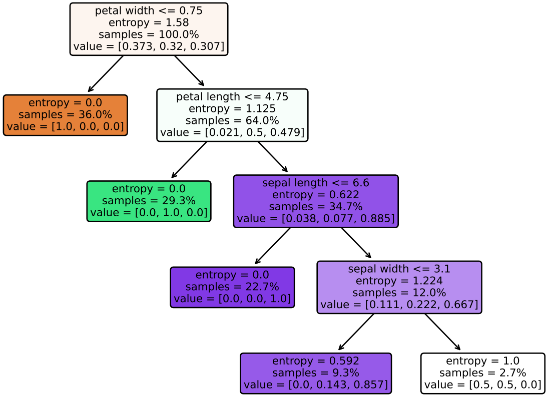
Build a new tree starting from each subset where the minority is non–empty



Recursion step

Observation

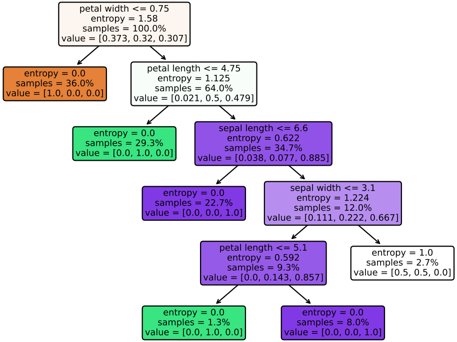
- The weighted sum of the entropy of the descendant nodes is always smaller than the entropy in the ancestor node, even if one of the descendant has higher entropy.
- Consider the three bottom right nodes (a=ancestor, ld=left descendant, rd=right descendant)



$$ent_a = 0.622 > (ent_{ld} * samp_{ld} + ent_{rd} * samp_{rd}) / samp_a = (0 * 22.7 + 1.224 * 12) / 34.7 = 0.39$$

Recursion ends

- Most of the leaves are pure, recursion impossible
- One of the leaves is not pure, but no more tests are able to give positive information gain, recursion impossible
    - it is labelled with the majority class, or, in case of tie, with one of the non–empty classes
- The error rate on the training set is 1.35%
    - 1 of the 75 examples in the training set is not correctly classified by the learned decision tree
    - it is one of the two items in the rightmost leaf



Building a *Decision Tree* with binary splits

procedure $\textsc{BuildTree}$(dataset $\mathcal{X}$, node p)
  if all the class values of $\mathcal{X}$ are c **then**
    return node p as a leaf, label of p is c
  if no attribute can give a positive information gain in $\mathcal{X}$ **then**
    say that the majority of elements in $\mathcal{X}$ has class c
    return node p as a leaf, label of p is c
  find the attribute d and threshold t giving maximum information gain in $\mathcal{X}$
  create two internal nodes descendant of p, say p_{left} and p_{right}
  let $\mathcal{X}_{left} =$ selection on $\mathcal{X}$ with $d < t$
  $\textsc{BuildTree}(\mathcal{X}_{left}, p_{left})$
  let $\mathcal{X}_{right} =$ selection on sdata with $d \geqslant t$
  $\textsc{BuildTree}(\mathcal{X}_{right}, p_{right})$

Decision tree for the Iris classifier

Internal representation

	ChLeft	ChRight	Feature	Threshold	NNodeSamples	Impurity
0	1	2	petal width	0.750000	75	1.579659
1	-	-	-	nan	27	0.000000
2	3	4	petal length	4.750000	48	1.124941
3	-	-	-	nan	22	0.000000
4	5	6	sepal length	6.600000	26	0.621904
5	-	-	-	nan	17	0.000000
6	7	10	sepal width	3.100000	9	1.224394
7	8	9	petal length	5.100000	7	0.591673
8	-	-	-	nan	1	0.000000
9	-	-	-	nan	6	0.000000
10	-	-	-	nan	2	1.000000

Training Set Error

- execute the generated decision tree on the training set itself
  - obviously the class attribute is hidden
- count the number of discordances between the true and the predicted class
- this is the *training set error*

Causes of non–zero training set error

https://app.wooclap.com/COFEKT

The training set error can be greater than zero because of...

Causes of non–zero training set error

https://app.wooclap.com/COFEKT

The training set error can be greater than zero because of...

- the limits of decision trees in general:
  - a decision tree based on tests on attribute values can fail
- insufficient information in the predicting attributes

Training Set Error

- Is this 1.35% interesting? What is its *meaning*?

Training Set Error

- Is this 1.35% interesting? What is its *meaning*?
- It is the error we make on the data we used to generate the classification model
- It is probably the lower limit of the error we can expect when classifying new data
- We are much more interested to an upper limit, or to a more significant value

Test set error

- The test set error is more indicative of the expected behaviour with new data
- Additional statistic reasoning can be used to infer error bounds given the test set error
- We have available 75 additional labelled records in the *Iris* dataset

Iris classification error

	Num Errors	Set Size	% *Wrong*
Training Set	1	75	1.35
Test Set	13	75	17.33

Iris classification error

	Num Errors	Set Size	$\%Wrong$
Training Set	1	75	1.35
Test Set	13	75	17.33

Why the test set error is so much worse?

Overfitting 1/2

Definition: overfitting happens when the learning is affected by *noise*

> *When a learning algorithm is affected by noise, the performance on the test set is (much) worse than that on the training set*

Overfitting 2/2

More formally

A decision tree is a *hypothesis* of the relationship between the predictor attributes and the class. Some definitions:

- h = hypothesis
- $error_{train}(h)$ = error of the hypothesis on the training set
- $error_{\mathcal{X}}(h)$ = error of the hypothesis on the entire dataset

h overfits the training set if there is an alternative hypothesis h' such that

$$\begin{aligned} error_{train}(h) &< error_{train}(h') \\ error_{\mathcal{X}}(h) &> error_{\mathcal{X}}(h') \end{aligned}$$

Causes for overfitting

1. Presence of noise
   - individuals in the training set can have bad values in the predicting attributes and/or in the class label, or can represent unusual cases
   - in this case, the model is influenced from partly wrong or unusual training data
2. Lack of representative instances
   - some situations of the real world can be underrepresented, or not represented at all, in the training set
   - this situation is quite common

A good model has low *generalization* error i.e. it works well on examples different from those used in training

Occam's Razor [5]

Everything should be made

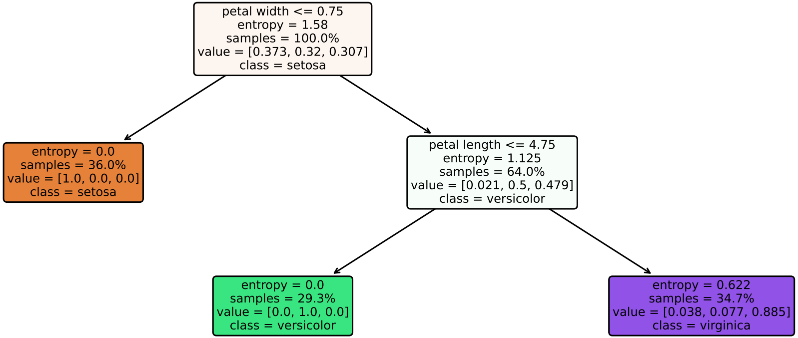
as simple as possibile,

but not simpler

- all other things being equal, simple theories are preferable to complex ones
- a long hypothesis that fits the data is more likely to be a coincidence
- pruning a decision tree is a way to simplify it
  - we need to find precise, quantitative guidelines for effective pruning

5 William of Ockham, an english franciscan philosopher of the 14-th century

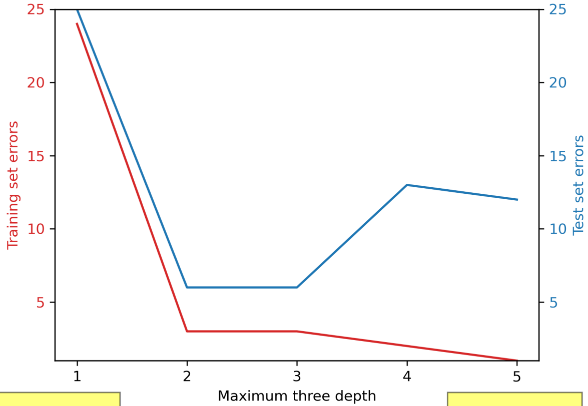
Example: Iris classification with pruned tree



	Num Errors	Set Size	% *Wrong*
Training Set	3	75	4.00
Test Set	6	75	8.00

General effect of model simplification

Pruning is the way to *simplify* the model when you are using a decision tree



High bias
few examples
correctly classified

more <==== pruning ====> less

High variance
noise has
high influence

Hyperparameters

- Every model generation algorithm can be adjusted by setting specific *hyperparameters*
- Each model has its own hyperparameters
- One of the hyperparameters of decision tree generation is the *maximum tree depth*

Choice of the attribute to split the dataset

- Looking for the split generating the maximum purity
- We need a measure for the purity of a node
  - a node with two classes in the same proportion has low purity
  - a node with only one class has highest purity

Impurity functions

Measures of the impurity of a node

Entropy – already seen [6]

Gini Index

Misclassification Error OPTIONAL

[6] it is available in *Scikit-Learn*

Gini Index[7] – Intuition

- Consider a node p with C_p classes
- Which is the frequency of the wrong classification in class j given by a random assignment based only on the class frequencies in the current node?
- For class j
  - frequency $f_{p,j}$
  - frequency of the other classes $1 - f_{p,j}$
  - probability of wrong assignment $f_{p,j} * (1 - f_{p,j})$
- the Gini Index is the total probability of wrong classification

$$\sum_j f_{p,j} * (1 - f_{p,j}) = \sum_j f_{p,j} - \sum_j f_{p,j}^2 = 1 - \sum_j f_{p,j}^2$$

7 This is the default impurity measure in *Scikit-Learn*

Gini Index – Discussion

- the maximum value is when all the records are uniformly distributed over all the classes: $\mathbf{1 - 1/}\boldsymbol{C_p}$
- the minimum value is when all the records belong to the same class: $\mathbf{0}$

Splitting based on the Gini Index

- Used by CART, SLIQ, SPRINT
- When a node p is split into ds descendants, say $p_1, \ldots, p_{ds}$
- Let $N_{p,i}$ and N_p be the number of records in the i-th descendant node and in the root, respectively
- We choose the split giving the maximum reduction of the Gini Index

$$GINI_{split} = GINI_p - \sum_{i=1}^{ds} \frac{N_{p,i}}{N_p} GINI(p_i)$$

Misclassification Error

Optional

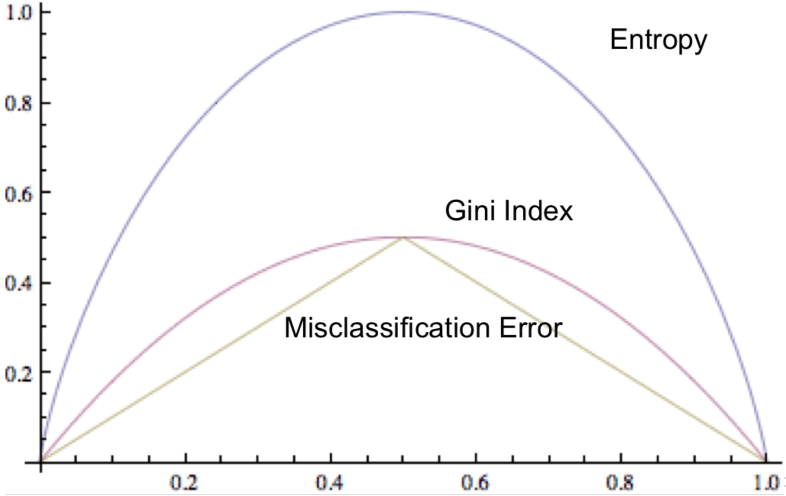
- If a node is a leaf, we find the highest label frequency; this frequency is the accuracy of the node and this label is the output of the node
- The misclassification error is the complement to 1 of the accuracy
- Since the most frequent class determines the node label, the complement is the error
  - The maximum value is when all the records are uniformly distributed over all the classes: $1 - 1/C_p$
  - The minimum value is when all the records belong to the same class: 0
- The choice of the split is done in the same way as for the Gini index

$$ME(p) = 1 - \max_j f_{p,j}$$

Comparison of the impurity functions

OPTIONAL

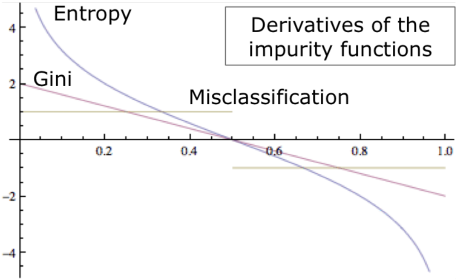
For two classes with frequencies f and $1 - f$



Comparison of the impurity functions – Discussion

Optional

- The behavior of ME is linear, therefore an error in the frequency is completely transferred into the impurity computation
- Entropy and Gini have varying derivative, with the minimum around the center
  - they are more robust w.r.t. errors in the frequency, when the frequencies of the two classes are similar



Complexity of DT induction (C4.5 algorithm) I

- N instances and D attributes in $\mathcal{X}$
  - tree height is $\mathcal{O}(\log N)$
- Each level of the tree requires the consideration of all the dataset (considering all the nodes)
- Each node requires the consideration of all the attributes
  - overall cost is $\mathcal{O}(DN \log N)$

Complexity of DT induction (C4.5 algorithm) II

- In addition
  - binary split of numeric attributes costs $\mathcal{O}(N \log N)$, without increment of complexity
  - pruning requires to consider globally all instances at each level, generating an additional $\mathcal{O}(N \log N)$, which does not increase complexity.

Characteristics of DT Induction I

1. It is a non–parametric approach to build classification models
   - it does not require any assumption on the probability distributions of classes and attribute values
2. Finding the best DT is NP–complete, the heuristic algorithms allow to find sub–optimal solutions in reasonable times
3. The run–time use of a DT to classify new instances is extremely efficient: $\mathcal{O}(h)$, where h is the height of the tree
4. Robust w.r.t. noise in the training set (i.e. wrong class labels), if the overfitting is avoided with appropriate pruning
5. Redundant attributes do not cause any difficulty

Characteristics of DT Induction II

- In case of strong correlation between two attributes, if one is chosen for a split, most likely the other will never provide a good increment of node purity, and will never be chosen

6. The nodes at a high depth are easily irrelevant (and therefore pruned), due to the low number of training records they cover
7. In practice, the impurity measure has low impact on the final result
8. In practice, the pruning strategy has high impact on the final result

Algorithms for building DTs

- Several variants, depending on
  - tree construction strategy
  - partition strategy
  - pruning strategy
- Tests based on linear combinations of numeric attributes
- Multivariate tests (e.g. $a = x$ and $b = y$)
- ...

Conclusion

- Decision trees are usually the best starting point to learn supervised machine learning
  - easy to understand
  - easy to implement
  - easy to use
- Overfitting can be controlled by adjusting the maximum tree depth
  - other adjustment are possible, depending on the implementation
    - e.g. lookup the
- Are able to predict discrete values (the class) on the basis of continuous or discrete predictor attributes[8]

8 The Scikit-Learn implementation of Decision Trees do not allow discrete attributes, therefore in these cases a *data transformation* is necessary

Important concepts

- Impurity functions: entropy, Gini, misclassification
- The recursive greedy algorithm for building a decision tree
- Training error and test error
- Why the test error can be much greater than the training error
- Why the pruning can improve the performance
- How to deal with continuous attributes

Questions

- Why maximising the Information Gain and the Gini Index gain should be, in general, better than minimising the Misclassification Error?
- Why do we prefer a greedy algorithm instead of trying all the possibile trees?
- Consider the Adult dataset. If the decision tree to predict wealth has the marital status near to the top, can we say that the marital status is a major cause for wealth?
- Can we say that the attributes which are not mentioned in the tree are not a cause for wealth?

Bibliography I

- Wray Buntine.
  Learning classification trees.
  Statistics and Computing, 2(2):63–73, Jun 1992.
  ISSN 1573-1375.
  doi: 10.1007/BF01889584.
  URL https://doi.org/10.1007/BF01889584.
- Earl B. Hunt, Janet Marin, and Philip J. Stone.
  Experiments in induction.
  Academic Press, 1966.
  URL https://books.google.it/books?id=sQoLAAAAMAAJ.
- Ross Quinlan.
  Discovering rules by induction from large collections of examples.
  In *Expert systems in the micro electronic age*. Edimburgh University Press, 1979.

Bibliography II

- Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus F. M. Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, and Dan Steinberg.
  Top 10 algorithms in data mining.
  Knowl. Inf. Syst., 14(1):1–37, 2008.
  doi: 10.1007/s10115-007-0114-2.
  URL http://dx.doi.org/10.1007/s10115-007-0114-2.